Práctica 1: Señales en MATLAB		Grupo	
		Puesto	
Apellidos, nombre		Fecha	
Apellidos, nombre			

El objetivo de esta práctica es presentar al alumno el modo de orientar las herramientas que ofrece MATLAB a la representación y manejo de señales y sistemas.

A partir del ejercicio 6, desarrolle cada ejercicio en un fichero de comandos 'ejercicio_X.m' separado (salvo cuando se le solicite desarrollar una función, en cuyo caso el fichero llevará el nombre de la función). Justo antes de finalizar la práctica, comprima los ficheros '.m' generados en un único fichero 'practica_1_Puesto_XX . zip', conéctese al sistema de entrega de prácticas de la Intranet y entréguelo en el grupo que corresponda

1.1 Generación y manipulación básica de señales

Para seguir este apartado escriba en la línea de comando todos los ejemplos mostrados. Utilice la ayuda de MATLAB para documentarse sobre cualquier comando que desconozca.

1.1.1 Ejercicio 1: representación de una señal en un rango dado

En general, una señal quedará representada por un vector fila o por un vector columna (es decir, por matrices con una única fila o columna). En MATLAB, todos los vectores se indexan comenzando por el **1**, es decir, **y(1)** es el primer elemento del vector **y**. Cuando este criterio no coincida con el del problema a resolver (e.g., porque el primer valor del vector y corresponda al índice **-5**), se puede crear un vector adicional de índices. Por ejemplo, para representar la señal:

$$x[n]=\begin{cases}2n, & -3\le 0\le 3\\ 0, & resto\end{cases}$$

se puede usar el operador '**:**' para definir un vector con los índices de **x[n]** no nulos, y luego definir el propio vector **x** de modo que contenga los valores deseados en cada uno de estos índices:

```
>> n=[-3:3];
>> x=2*n;
```

Represente esta señal escribiendo **stem(n,x)**. Para examinar la señal en un rango más amplio de índices, será necesario extender tanto el vector de índices, **n**, como la señal **x**:

— Para representar la señal en el intervalo **[-5,5]**:

```
>> n=[-5:5];
>> x=[0 0 x 0 0];  % x tenía el valor del ejemplo anterior
```

— Para representarla en **[-100,100]**:

```
>> n=[-100:100];
>> x=[zeros(1,95) x zeros(1,95)];  % x tenía el valor del ejemplo anterior
```

Represente cada una de estas tres señales en tres figuras distintas (vea el comando **figure**).

1.1.2 Ejercicio 2: representación de dos señales en un cierto rango

Sean $x_1[n]=\delta[n]$ y $x_2[n]=\delta[n+2]$ (la función $\delta[n]$ toma valor 1 para *n=0* y valor nulo en el resto). Dibuje aparte el aspecto que tienen ambas señales. Estas señales pueden definirse en MATLAB escribiendo:

```
>> nx1=[0:10];
>> x1=[1 zeros(1,10)];
>> nx2=[-5:5];
>> x2=[zeros(1,3) 1 zeros(1,7)];
```

Para representarlas, basta escribir **`stem(nx1,x1)`** y **`stem(nx2,x2)`**. Compruebe que obtiene el resultado esperado.

Represéntelas ahora directamente con **`stem(x1)`** y **`stem(x2)`**, función que en ausencia de un vector de índices asume que éste comienza en **1** y que tiene la misma longitud que la señal. Indique, en esta situación, cual es la expresión analítica de las señales que observa:

Señal representada con **`stem(x1)`**	
Señal representada con **`stem(x2)`**	

Tenga en cuenta en lo sucesivo que a la hora de representar señales, tan importante como la expresión de la señal es el vector de índices con respecto al cual se representa.

1.1.3 Ejercicio 3: representación de señales continuas

Una señal continua es posible representarla mediante vectores que contengan valores de dicha señal en instantes de tiempo muy cercanos entre sí. Así, si se quiere representar una señal continua en el intervalo $-5 \leq t \leq 5$ mediante la expresión de un valor cada *0.1* segundos, tenemos dos opciones para crear el vector de *índices* (en este caso instantes de tiempo):

```
>> t=[-5:0.1:5];
```

, o bien:

```
>> t=linspace(-5,5,101);
```

Hecho esto, para representar la señal $x(t)=sen(\pi t/4)$ basta con escribir:

```
>> x=sin(pi*t/4);
```

Observe que en MATLAB, cuando el argumento de una función de este tipo (*sin*, *cos*, *exp*, etc.) es un vector, el resultado es un vector del mismo tamaño, en el que cada valor resulta de la aplicación de la función a cada valor del vector argumento. Para representar gráficamente la señal, resaltando su carácter de señal continua, utilice **`plot`** en vez de **`stem`**:

```
>> plot(t,x);
```

Represente gráficamente las señales $x_1(t) = sen(\pi t/4)$ y $x_2(t) = \cos(\pi t/4)$ en el intervalo $-4 \le t \le 4$ dando valores cada 1/8 de segundo. Represente ambas sobre la misma figura utilizando el comando **`plot`** y, a continuación, nuevamente sobre la misma figura, represente ambas con el comando **`stem`** (para ello utilice el comando **`hold`**). Utilice dos colores: uno para las dos representaciones de $x_1(t)$ y otro para las dos de $x_2(t)$.

En lo sucesivo, cada siempre que se quiera representar gráficamente una señal de tiempo discreto utilice el comando **`stem`**; análogamente, siempre que la señal sea de tiempo continuo (aunque con MATLAB se aproxime por una señal de tiempo discreto definida a intervalos regulares y *muy pequeños*) utilice el comando **`plot`** para resaltar este hecho y evitar cualquier confusión.

1.1.4 Ejercicio 4: representación de señales complejas

Sea ahora el caso de una exponencial compleja discreta $x[n] = e^{j(\pi/8)n}$ en el intervalo $0 \le n \le 32$:

```
>> n=[0:32];
>> x=exp(j*(pi/8)*n);
```

El vector **`x`** contiene una serie de *33* valores complejos de la señal $x[n]$. Represéntelos gráficamente, haciendo uso de la función **`stem`**, indicando qué característica de cada valor complejo desea representar:

```
>> stem(n,real(x));
>> stem(n,imag(x));
>> stem(n,abs(x));
>> stem(n,angle(x));
```

Compruebe y recuerde que si en la función no se especifica qué característica de la señal compleja se desea representar (es decir, si escribe **`stem(n,x)`**), MATLAB representará, por defecto, la parte real de la señal y mostrará una advertencia en la línea de comando indicándolo.

1.1.5 Ejercicio 5: operaciones aritméticas con señales

Siempre que dos señales compartan el mismo vector de índices (es decir, que el vector que representa cada señal tenga el mismo origen de tiempos), es posible realizar directamente cierto tipo de operaciones básicas. Así, defina las señales:

```
>> x1=sin((pi/4)*[0:30]);
>> x2=cos((pi/7)*[0:30]);
```

y efectúe las siguientes operaciones:

```
>> y1=x1+x2;
>> y2=x1-x2;
>> y3=x1.*x2;
>> y4=x1./x2;
>> y5=2*x1;
```

Observe que en el caso de la multiplicación y división, es necesario preceder el operador de un punto, para indicar que la operación ha de llevarse término a término, en vez de entre matrices (e.g., el producto de matrices requiere que el segundo término tenga tantas filas como columnas tenga el primero, algo que no verifican los vectores **x1** y **x2**).

Represente las siete señales de este apartado en siete figuras distintas.

1.1.6 Ejercicio 6: *scripts* y funciones

En MATLAB hay esencialmente dos tipos de ficheros con extensión '.m': *scripts* de comandos y

funciones. Su uso es imprescindible de cara a organizar, depurar y guardar los ejercicios (en *scripts*) y siempre que se requiera realizar un mismo conjunto de operaciones (es decir, funciones) sobre señales diferentes.

Tenga en cuenta que para poder invocar *scripts* y funciones, los ficheros '.m' que los implementan han de estar en algún lugar referenciado por el *path* de MATLAB. Utilice algún directorio de su unidad de disco privada (por defecto, **h:**\) y añádalo al *path* de MATLAB (menú '*File/Set path…*'). Recuerde que tendrá que efectuar esta operación cada vez que reinicie su ordenador.

Replique los ejemplos que se presentan a continuación:

Genere con el editor de MATLAB el siguiente *script* (asígnele el nombre 'ejercicio6.m'), cuyo objetivo es representar una determinada señal discreta en un intervalo dado, calcular su valor medio en el citado intervalo, y representar este valor como una función constante:

```
% ejercicio6.m
n = [0:16];
xl = cos(pi*n/4);
yl = mean(xl);
stem(n,xl)
title('xl = cos(pi*n/4)')
xlabel('n (samples)')
ylabel('xl[n]')
hold on
m1=y1*ones(1,17);
plot(n,m1)
hold off
```

Invoque el *script* creado escribiendo `ejercicio6` en la línea de comandos de MATLAB.

Si ahora desea realizar la misma operación con la señal $x_2[n] = \sin(\pi n/4)$ y en el intervalo $0 \le n \le 32$, basta con copiar el fichero, asignarle un nuevo nombre (e.g., 'ejercicio6b.m') y cambiar las líneas que proceda:

```
% ejercicio6b.m
n = [0:32];
x2 = sin(pi*n/4);
y2 = mean(x2);
stem(n,x2)
title('x2 = sin(pi*n/4)')
xlabel('n (samples)')
ylabel('x2[n]')
hold on
m2=y2*ones(1,33);
plot(n,m2)
hold off
```

Compruebe que obtiene el resultado deseado ejecutando este segundo *script* desde la línea de comandos.

Un fichero '.m' también puede representar una función. Para ello, la primera palabra del fichero ha de ser `function`. El resto de la línea especifica los parámetros que acepta la función y los valores que devuelve. El siguiente ejemplo muestra una función llamada `f_obtiene_yz`, que toma como parámetro un vector `x`, y devuelve otros dos vectores, `y` y `z` (acostúmbrese a preceder el nombre de todas las funciones con las letras `f_`, de modo que se distingan claramente de funciones de MATLAB o de *scripts*):

```
function [y,z] = f_obtiene_yz(x)
% [y,z] = f_obtiene_yz(x) admite una señal 'x' y
% devuelve dos señales, 'y' y 'z', donde 'y' vale 2*x
% y 'z' vale (5/9)*(x-3)
y = 2.*x;
z = (5/9).*(x-3);
```

El siguiente ejemplo muestra cómo utilizar esta función desde la línea de comandos (o bien desde un *script*):

```
>> n=[0:15];
>> x1=4*sin((pi/4)*n);
>> [y1,z1]=f_obtiene_yz(x1);
>> stem(n,x1);
>> hold on;
>> stem(n,y1,'r');
>> stem(n,z1,'g');
>> hold off;
```

1.2 Operaciones con señales

Realice todos los ejercicios que se le solicite en ficheros '.m', de modo que pueda guardar y modificar sus resultados sin necesidad de volver a teclear el código de nuevo. Después de cada ejercicio, solicite al profesor de prácticas que valide el resultado en su memoria de prácticas.

1.2.1 Ejercicio 7: transformaciones de la variable independiente

Defina en un fichero la siguiente función discreta, $x[n]$, en el intervalo $-3 \le n \le 11$, a través de un vector **x** y del vector de índices **nx** correspondiente:

$$x[n] = \begin{cases} 2, & n = 0 \\ 1, & n = 2 \\ -1, & n = 3 \\ 3, & n = 4 \\ 0, & resto \end{cases}$$

Represéntela gráficamente y fije las etiquetas necesarias de modo que el resultado sea similar al que muestra la Fig. 1.

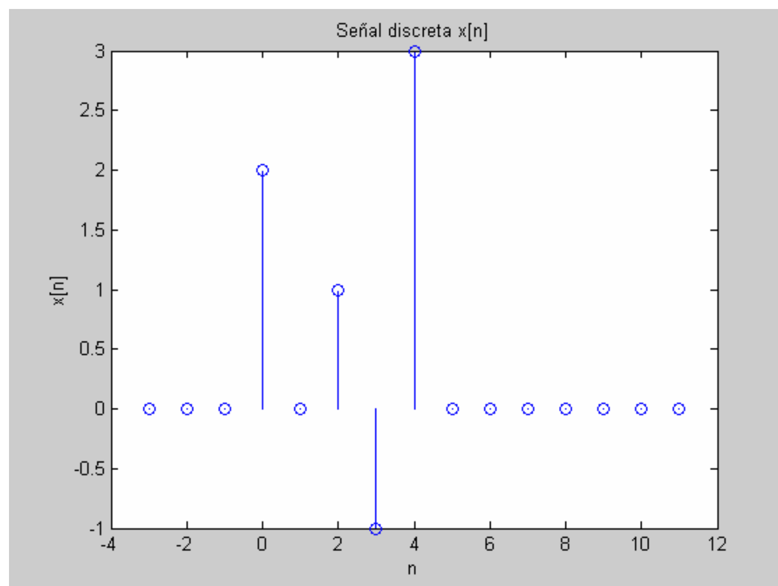


Fig.1: Representación gráfica de la señal original

Una vez definida la señal $x[n]$, defina en el mismo fichero '.m' las siguientes señales:

$$y_1[n] = x[n-2]$$
$$y_2[n] = x[n+1]$$
$$y_3[n] = x[-n]$$
$$y_4[n] = x[-n+1]$$

Para ello, el método que se propone, orientado a evidenciar que no cambia la señal sino la variable independiente, consiste en primero definir `y1=x`, `y2=x`, etc., y a continuación definir los correspondientes vectores índice de cada señal (`ny1`,…, `ny4`) como una transformación del vector de índices `nx`. Para ello dibuje aparte las cuatro señales que se le solicitan, deduzca la relación que existe entre sus respectivos vectores de índices y el vector `nx`, y exprese dicha relación mediante MATLAB.

Finalice este apartado representando gráficamente la señal original y, en otra ventana, las cuatro señales resultantes de cada transformación (ver el comando `subplot`); titule y etiquete cada gráfico. Indique en cada uno cuál es la relación entre la señal representada $y_i[n]$ y la original $x[n]$ (e.g., "adelantada tres posiciones", "invertida y luego retrasada…").

1.2.2 Ejercicio 8: señales pares e impares

Para cada una de las señales cuya expresión se ofrece a continuación, obtener su parte par y su parte impar (para generar las señales invertidas utilice `fliplr` en vez de modificar la variable independiente o vector de índices). Representar a continuación las tres señales en una misma figura (utilice `subplot`), utilizando para todas el rango $-10 \le n \le 10$. Codifique todo el ejercicio en un mismo *script*.

$$x_1[n] = \begin{cases} 1, & 0 \le n \le 5 \\ -1, & -5 \le n < 0 \\ 0, & resto \end{cases}$$

$$x_2[n] = \begin{cases} 1, & n = -2 \\ 2, & n = -1 \\ 3, & n = 0 \\ 1, & n = 7 \\ 0, & resto \end{cases}$$

$$x_3[n] = \begin{cases} -1, & n \in \{-4,3\} \\ 2, & n \in \{-3,-2,1\} \\ 1, & n \in \{-1,0,2\} \\ 0, & resto \end{cases}$$

Compruebe que el ejercicio es correcto observando las simetrías de las señales pares e impares obtenidas. Observe que si se desea efectuar operaciones aritméticas con varias señales distintas (caso de este ejercicio), todas ellas deben compartir el mismo vector de índices; por lo tanto, las transformaciones de la variable independiente han de trasladarse en este caso a la señal (no como en el ejercicio anterior).

Cree una función `f_descompone_par_impar` que tome como parámetro una señal y devuelva dos señales: su parte par y su parte impar; copie el *script* anterior y modifíquelo de modo que haga uso de la función implementada (y que será llamada tantas veces como señales tiene este ejercicio).